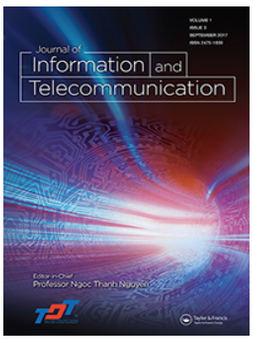

Taylor & Francis
Taylor & Francis Group

Journal of Information and Telecommunication

ISSN: 2475-1839 (Print) 2475-1847 (Online) Journal homepage: https://www.tandfonline.com/loi/tjit20

Analysis of temporal complex events in sensor networks

Juan Boubeta-Puig, Mario Bravetti, Luis Llana & Mercedes G. Merayo

To cite this article: Juan Boubeta-Puig, Mario Bravetti, Luis Llana & Mercedes G. Merayo (2017) Analysis of temporal complex events in sensor networks, Journal of Information and Telecommunication, 1:3, 273-289, DOI: 10.1080/24751839.2017.1347763

To link to this article: https://doi.org/10.1080/24751839.2017.1347763

Published online: 18 Jul 2017.

JOURNAL OF INFORMATION AND TELECOMMUNICATION, 2017
VOL. 1, NO. 3, 273–289
https://doi.org/10.1080/24751839.2017.1347763

Analysis of temporal complex events in sensor networks

Juan Boubeta-Puig [a], Mario Bravetti[b], Luis Llana[c] and Mercedes G. Merayo [c]

[a]Department of Computer Science and Engineering, University of Cádiz, Cádiz, Spain; [b]Dipartimento di Informatica – Scienza e Ingegneria, Università di Bologna, Bologna, Italy; [c]Departamento de Sistemas Informáticos y Computación, Universidad Complutense de Madrid, Madrid, Spain

ABSTRACT

In this paper we introduce a framework for detecting anomalies in the clocks of the different components of a network of sensor stations connected with a central server for measuring of air quality. Local clocks of sensor stations can be advanced/delayed with respect to the central server clock and this situation provokes the inaccuracy in the interpretation of the collected data. We propose a novel approach, supported by a formal representation of the network using fuzzy-timed automata, to precisely represent the expected behaviour of each component of the network. Using fuzzy logic concepts, we can specify admissible mismatches between the clocks. In addition, we apply complex event processing (CEP) technology in order to automatically detect situations of interest while processing the massive amount of data transferred across the network. Specifically, we have designed a collection of CEP patterns that trigger alarms when unexpected differences are observed. We also report the results obtained from the application of our approach to the network during December 2016.

ARTICLE HISTORY
Received 31 March 2017
Accepted 12 April 2017

KEYWORDS
Real-time automatic analysis; CEP; fuzzy logic; sensor networks

1. Introduction

The growth of smart cities is increasing the use of sensor networks to control how the citizens might be affected by different hazards. In particular, sensor networks providing air quality information are highly demanded because air pollution is a major environmental health problem. In fact, air quality affects everyone in developed and developing countries (WHO, 2016), worsening the effects of certain illnesses and even causing the death of specific risk groups. Even though there exist several systems providing air quality information to the citizens, there is a need to *intelligently* collect and process the information of different sensors conforming a network. However, these systems present a main problem: the collection of data can include wrong information, providing misleading judgements. In particular, the information concerning time can be wrong due to the devices measuring the passing of time or due to delays produced by the overloading of the communication network or by a bottleneck in the server. The consequences of a wrong pattern recognition process are, among others, incongruities checking timing restrictions, bad decisions derived from the collected results and alarms generated at untimely moments.

CONTACT Mercedes G. Merayo mgmerayo@fdi.ucm.es

Therefore, given the fact that determining clocks accuracy is quite difficult, we need to use a mechanism to manage time constraints that must be fulfil by the systems. Suppose that a certain signal should be received every 10 min and there is a lapse of 10 min and 0.001 s since the reception of the last message. In this scenario, we should not raise an alarm because this *error* might be due to several reasons, none of them showing a failure of the device sending the signal. For example, the clock of the receiver might be wrong or the message might have been delayed while being retransmitted in a point between the sender and the receiver.

We advocate that in order to precisely analyse the correctness of a system, we should have a *blueprint* (Lamport, 2015). We have used *fuzzy-timed automata* to specify the expected behaviour of sensor networks and, therefore, detect anomalies in the real networks. In this paper we distinguish two types of anomalies: (1) *malfunction*, when a station clock is delayed or advanced between 5 and 10 min with respect to the clock server and (2) *failure*, when a station clock is advanced/delayed more than 10 min with respect to the clock server. The idea of using fuzzy logic in the field of automata theory is not new. It has been used in the past to represent imprecise specifications (Mensch & Lipp, 1990) and to model learning systems (Wee & Fu, 1969). Fuzziness has been introduced in the different components of the automata: states, transitions, and actions (Doostfatemeh & Kremer, 2006; Mraz, Lapanja, Zimic, & Virant, 1999; Wee & Fu, 1969). Although there are many proposals to define fuzzy automata (Andrés, Llana, & Núñez, 2011; Doostfatemeh & Kremer, 2005; Mordeson & Malik, 2002; Wee & Fu, 1969), we considered timed automata because it is much easier to reuse its existing tools to deal with fuzzy time than build tools from scratch. Technically, our proposal is a hybrid between timed automata (Alur & Dill, 1994) and our proposal of fuzzy automata (Andrés et al., 2011) and it has similarities with previous work (Fernández-Vilas, Pazos-Arias, & Díaz Redondo, 2002), although we use fuzzy logic instead of many valued logic. Finally, it is worth to point out that our formalism specifically focusses on the representation of time information. Therefore, we designed the language to concentrate on features that are difficult, or even impossible, to represent by using general purpose fuzzy automata models.

The second major component of our approach is to use a pattern recognition process so that better quality information can be provided. In this line, the complex event processing (CEP) technology (Luckham, 2012) together with fuzzy logic can play a fundamental role, as we will show in this paper. CEP allows us to process and correlate huge amounts of data with the aim of detecting critical or relevant situations in real time. In addition, fuzzy reasoning allows us to deal with imprecision in the collected information.

The main goal of this paper is to show our experiences and conclusions after analysing a fully deployed air sensors network in order to automatically detect potential (temporary) malfunctions and errors in the measure of time. Specifically, we consider the network conformed by a central server connected to 61 sensor stations, each of them receiving data from 6 sensors, devoted to the capture of different air pollutants around Andalusia (a region in the South of Spain). We have used real-time data, observed during December 2016, and have focused on assessing the accuracy of the clocks of each station because this is a key factor for the correct performance of the event patterns applied by the CEP engine located in the central server.

The rest of the paper is structured as follows. In Section 2 we introduce the main concepts of CEP technology. Section 3 describes our fuzzy-timed automata model for detecting anomalies in sensor networks. Sections 4 and 5 present our case study over a sensor network for controlling air quality and the analysis of the results obtained, respectively. Finally, we give our conclusions and future work in Section 6.

2. Complex event processing

Complex event processing (Luckham, 2012) is a cutting-edge technology that allows us to analyse and correlate vast amounts of data in form of events with the aim of detecting situations of interest in real time.

An event can be defined as anything that happens or could happen (Luckham, 2002), but also anything that could happen but does not happen. A situation is an event occurrence or an event sequence that requires an immediate reaction (Etzion & Niblett, 2011). A simple event is indivisible and happens at a point in time, while a complex event contains more semantic meaning which summarizes a set of other events (Event Processing Glossary - Version 2.0, 2011). Complex events can be derived from other events by applying event patterns, templates where the conditions describing situations to be detected are specified. A CEP engine is the software used to match these patterns over continuous and heterogeneous event streams (timely ordered sequence of events), and to raise alerts about complex events created when detecting such event patterns.

Figure 1 depicts the event pattern recognition in CEP. It is performed in three stages:

(1) Event capture: it consists of the reception of events to be analysed by CEP technology.
(2) Analysis: from the event patterns previously defined in the CEP engine, it will process and correlate the information in the form of events in order to detect situations of interest in real time.
(3) Response: after detecting a particular situation, it will be notified to the system, software or device in question.

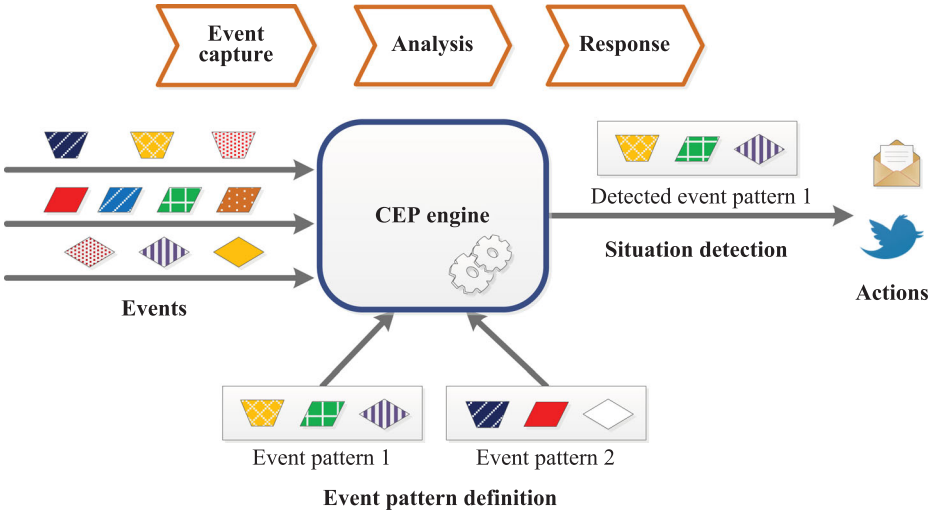

Figure 1. Event pattern recognition in CEP.

The main advantage of using CEP to process complex events is that they can be identified and reported in real time, unlike traditional software for event analysis, therefore reducing the latency in decision making. Other relevant advantages are: decision quality improvement, faster and automatic reply, information overload prevention and human workload reduction.

Therefore, CEP is a fundamental technology for applications that must produce a fast reply to situations that change quickly and asynchronously, must quickly react to unusual situations, and require loose coupling and adaptability (Chandy & Schulte, 2010).

3. Fuzzy-timed automata

In this section we present our *fuzzy* version of timed automata for specifying all the components of the network. Essentially, we use a type of finite automata where transitions are labelled with an action, as usual, and a fuzzy constraint to ensure that the collected data fulfils the expected properties. First, we introduce some basic concepts of fuzzy logic that will be used afterwards in the definition of our fuzzy-timed automata.

In ordinary logic, a set or a relation is determined by its characteristic function: a function that returns true if the element is in the set (or if some elements are related) and false otherwise. In the fuzzy framework, we have a complete range of values in the interval [0, 1]; the larger is the value, the more confidence we have in the assessment. We consider relations over the set of real numbers **R**. Therefore, a fuzzy relation is a mapping from the Cartesian product $\mathbf{R}^n$ into the interval [0, 1].

In this paper, we consider the fuzzy relations shown in Figure 2. We use these functions to define constraints in fuzzy-timed automata. These relations depend on a non-negative real number $\lambda \geq 0$. Given two values x and y each relation returns a value that represents a level of confidence in the assessment of the corresponding expression ($x = y$, $x \leq y$ and $x \geq y$) taking into account a specific threshold λ. These functions will allow us to consider some imprecision in the temporal behaviour of the systems.

$$\overline{x = y}^{\lambda} = \begin{cases} 0 & \text{if } x \leq y - \lambda \text{ or } x > y + \lambda \\ 1 + \frac{x-y}{\lambda} & \text{if } y - \lambda < x \leq y \\ 1 - \frac{x-y}{\lambda} & \text{if } y < x \leq y + \lambda \end{cases}$$

$$\overline{x \leq y}^{\lambda} = \begin{cases} 1 & \text{if } x < y \\ 0 & \text{if } x > y + \lambda \\ 1 - \frac{x-y}{\lambda} & \text{if } y \leq x \leq y + \lambda \end{cases}$$

$$\overline{x \geq y}^{\lambda} = \begin{cases} 1 & \text{if } x > y \\ 0 & \text{if } x \leq y - \lambda \\ 1 + \frac{x-y}{\lambda} & \text{if } y - \lambda < x \leq y \end{cases}$$

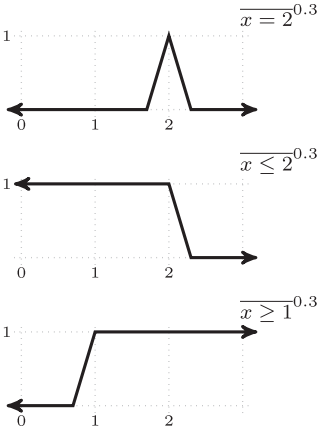

Figure 2. Fuzzy order relations.

A triangular norm (abbreviated *t*-norm) is a binary operation used in fuzzy logic to generalize the *conjunction* in propositional logic. Therefore, we require a *t*-norm to satisfy similar properties. We also require an extra property: monotonicity. Intuitively, the resulting truth value does not decrease if the truth values of the arguments increase.

Definition 3.1: A t-*norm* is a function $T : [0, 1] \times [0, 1] \mapsto [0, 1]$ which satisfies the following properties:

- Commutativity: $T(x, y) = T(y, x)$.
- Monotonicity: $T(x, y) \leq T(z, u)$ if $x \leq z$ and $y \leq u$.
- Associativity: $T(x, T(y, z)) = T(T(x, y), z)$.
- Number 1 is the identity element: $T(x, 1) = x$.
- Number 0 is nilpotent: $T(x, 0) = 0$.

Since *t*-norms are associative, we can generalize them to take as parameter a list of values:

$$T(x_1, x_2, \ldots, x_{n-1}, x_n) = T(x_1, T(x_2, \ldots, T(x_{n-1}, x_n)\ldots)).$$

The following *t*-norms are often used:

Łukasiewicz t-*norm*: $T(x, y) = \max(0, x + y - 1)$. We represent this *t*-norm with the symbol $\curlywedge$.
Gödel t-*norm*: $T(x, y) = \min(x, y)$. We represent this *t*-norm with the symbol $\bar{\wedge}$.
Product t-*norm*: $T(x, y) = x \cdot y$ (real number multiplication). We represent this *t*-norm with the symbol ★.
Hamacher product t-*norm*: $T(\delta_1, \delta_2) = \delta_1 \cdot \delta_2/(\delta_1 + \delta_2 - \delta_1 \cdot \delta_2)$. We represent this *t*-norm with the symbol ⋇.

In order to define fuzzy-timed automata, we need some additional elements. First, we need a set of *variables*. We denote the variables by $x, y, z, \ldots$. These variables will take values in real numbers. A tuple of variables $(x_1, x_2 \ldots, x_n)$ will be denoted by $\bar{x}$.

Definition 3.2: Given a set X of variables, an *environment* over X is a function $e : X \mapsto \mathbf{R}$ that assigns real numbers to every variable in X. The set of all the functions from X to $\mathbf{R}$ is denoted by Env_X.

In classical timed automata theory (Alur & Dill, 1994), time is expressed in the time constraints. Hence, we need to modify these constraints in order to be able to introduce fuzziness. In ordinary timed automata theory, the time constraints consist of conjunctions of inequalities. Instead of ordinary crisp inequalities, we use their fuzzy counterparts appearing in Figure 2. We could have more freedom in allowing general convex fuzzy sets, but we have preferred to keep our constraints close to the original ones so we can use the theory developed for timed automata. The role of a conjunction in Fuzzy Theory is played by *t*-norms. There is not a *canonical* *t*-norm: we have presented 4 of the more used *t*-norms and the designer can specify which one is more appropriate in each situation.

Definition 3.3: Given a set X of variables, the set FC_X of *fuzzy constraints* over X is defined by the following B.N.F.:

$$C ::= \text{True} \mid C_1 \triangle C_2 \mid \overline{x \bowtie n}^{\lambda} \mid \overline{x - y \bowtie n}^{\lambda},$$

where $\triangle$ is a t-norm, $\bowtie \in \{\leq, =, \geq\}$, $x, y \in X$, $\lambda \in \mathbf{R}^+$, and $n \in \mathbf{N}$.

In timed automata theory, constraints are used to decide if the automata can stay in a location and to decide if a transition can be executed. All this is done by checking if the values of the clocks satisfy the corresponding constraint. In fuzzy theory, the notion of satisfaction is not crisp: we do not have a boolean answer but a value in the interval $[0, 1]$. Therefore, we do not have that a constraint is true or false but a *satisfaction grade* of a constraint.

Definition 3.4: Let X be a set of variables, e be an environment over X and $C \in FC_X$ be a fuzzy constraint. We inductively define the *satisfaction grade of* C *in* e, written $\mu_C(e)$, as

$$\begin{array}{cc} 1 & \text{if } C = \text{True}, \\ \overline{e(x) \bowtie n}^{\lambda} & \text{if } C = \overline{x \bowtie n}^{\lambda},\ \bowtie \in \{\leq, =, \geq\}, \\ \overline{e(x) - e(y) \bowtie n}^{\lambda} & \text{if } C = \overline{x - y \bowtie n}^{\lambda},\ \bowtie \in \{\leq, =, \geq\}, \\ \triangle(\mu_{C_1}(e), \mu_{C_2}(e)) & \text{if } C = C_1 \triangle C_2. \end{array}$$

Let us remark that $\mu_C(e) \in [0, 1]$.

We consider a set of labels denoted by A and a set of variables X. Based on these sets, we define the sets of input and output actions:

- Input: action that implies the reception of data. They are identified by the symbol ? and they are defined by an action, the id of the sender and the tuple of variables in which the information will be stored. An example of these actions is the following: $read?(ids, (x_1, x_2 \cdots x_n))$. Formally, $A?$ is the set

$$\left\{ a?(ids, \bar{x}) \mid a \in A, ids \in \mathbf{N},\ \bar{x} \in \bigcup_{n=1}^{|X|} X^n \right\}.$$

- Output: action that implies the sending of data. They are identified by the symbol ! and they are defined by an action, the *id* of the sender, the id of the receiver and the information that is sent. An example of these actions is the following: $write!(id_s, id_r, (inf_1, inf_2 \cdots inf_k))$. Formally, $A!$ is the set

$$\left\{ a!(ids, idr, \bar{v}) \mid a \in A, ids, idr \in \mathbf{N},\ \bar{v} \in \bigcup_{n=1}^{|X|} \mathbf{R}^n \right\}.$$

It is worth noting that the tuples of variables/values that are included in the input/output actions can have different length. Finally, we consider the set $Acts = A! \cup A?$.

Now, we introduce a formal syntax to define fuzzy-timed automaton and network of automata. The operational semantics of automata networks is a little bit involved an it is out of the scope of this paper (Bouteta-Puig, Camacho, Llana, & Núñez, 2017).

Definition 3.5: A *fuzzy-timed automaton* is a tuple $(L, l_0, id, E, I, Acts, X)$ where:

- L is a finite set of locations.
- $l_0 \in L$ is the initial location.
- $id \in \mathbf{N}$ is the identifier of the automata.
- $E \subseteq L \times Acts \times FC_X \times L$ is the set of edges; we write $l \xrightarrow{a, C} l'$ whenever $(l, a, C, l') \in E$.
- $I : L \mapsto FC_X$ is a function that assigns constraints to locations.
- X is a finite set of variables.

An *automata network* is given by the following B.N.F.

$$N ::= A_f \mid \|_S(N_1, \ldots, N_k),$$

where $k \geq 2$ is a natural number, $S \subseteq A$ is the synchronization alphabet, and A_f is a fuzzy automaton.

Intuitively, an automata network is a collection of automata synchronizing in the actions belonging to the synchronization set S.

In Sections 2 and 3, we have argued the usefulness of CEP and presented a formalism to represent systems where time information can be handle with inaccuracy. In the next section, we will see how these apparently orthogonal theories can work together to analyse and improve the behaviour of complex systems. On the one hand, if we have a formal representation of the system that we would like to analyse, then it is easier to decide whether the system is presenting an unexpected (probably faulty) behaviour. In our case, this formal model will be given by a network of automata, where the expected behaviour of each component of the system is formally represented by an automaton. On the other hand, the use of CEP facilitates the task of processing raw information collected from the system that we are analysing. In our case, our patterns focus on detecting malfunctions of the clocks of the different components of the network.

4. Case Study: a sensor network controlling air quality

Next we provide a full description of the sensor network and the procedure that we have used in the evaluation of our approach. The air sensor network for controlling air quality that we have utilized belongs to the Andalusian regional government. The total area of the region is 87,268 Km^2 (33,694 square miles) and it currently has a population of around 8.4 million people. As previously mentioned, this type of wide sensor networks can present problems when managing data coming from several sensor stations if the station clocks are not synchronized. To address this challenge, we use a fuzzy logic method for processing sensor time delays and failures in this network. Moreover, we implement a set of event patterns based on this fuzzy logic method to automatically detect malfunctions and failures in the clocks. We put into practice our approach by using real-time data observed during December 2016.

This network is composed of several sensor stations. Every station gathers the air pollutant measurements taken from the eight Andalusian provinces. Then, this information is sent to a main server by GPRS or Internet. The data sent to the server is stored in a database and published in a web site. By using a web form, users can ask for downloading air data measured during a specific period of time; this information can be retrieved in PDF or HTML formats. However, this is a strong limitation for those that would like to process the information in real time as well as integrating it with third-party systems. To partially

address this need, the Andalusian regional government is sending the data to a dedicated server located in our university every 10 min.

The network is composed of a set of sensors for measuring pollution that belong to the Andalusian government, together with other sensors whose own are other public and private enterprises. These sensors are located in representative zones with the aim of optimizing the information about pollution spatial distribution. Some of them are located in zones in which sensor readings are not too influenced by local conditions, while others are located in zones where local conditions may impact on sensor measurements, such as road traffic pollution. Figure 3 shows how sensors have been located around the Andalusian territory.

Depending on the particularities of every zone, stations have more or less sensors in charge of measuring some of the following pollutants: particles matter smaller than 10 or 2.5 microns, sulfur dioxide, nitrogen oxides, carbon monoxide, ozone, benzene, toluene, xylene, ethylbenzene and hydrogen sulphide. In addition, other sensors are capable of measuring some meteorological elements: wind, precipitation, moisture, solar radiation, pressure, and temperature.

It is worth noting that we are only considering the key air pollutants: $PM_{2.5}$, PM_{10}, CO, O_3, NO_2 and SO_2; the other ones are not relevant for our study's purpose. For this reason, although the whole network is composed of 91 air sensor stations (646 sensors, 86.200 measurements/day) and 12 meteorological towers (231 sensors, 3.400 measurements/day), we only make use of 61 sensor stations.

Following, we present the specification of the air sensor network using fuzzy-timed automata. Next, we will introduce the methodology that we have applied to detect malfunctions and failures in sensor stations.

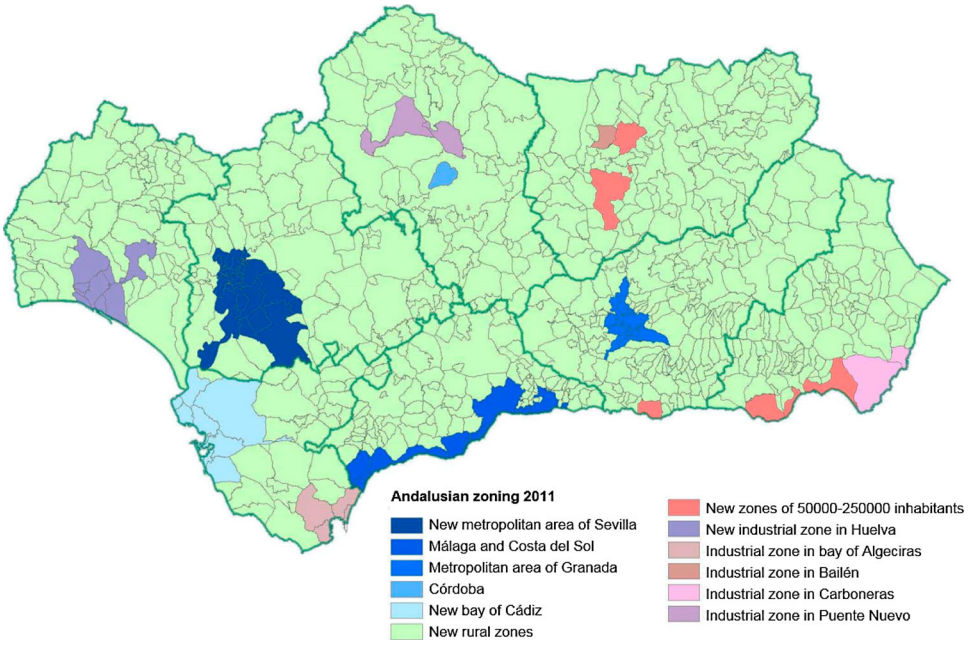

Figure 3. Map of air quality zoning in Andalusian territory.

4.1. Network specification

The air sensor network presents three kinds of entities: sensors, sensor stations, and a central server. *Sensors* are the small devices collecting information about a particular pollutant. Every sensor is associated with a *sensor station* with the aim of sending the data that they capture. The sensor stations are connected to 6 sensors and a *central server*. After processing the information, every 10 min, the stations send the collected data to the server. If the information is not sent, the station will inform the server about an error in order to be reset. The sensor stations also store the summary of timing results sent by the server and generated by the CEP engine, with the aim of having a record of the advances/delays that the stations present. The last entity is the *central server*, connected to 61 sensor stations that conform the Andalusian sensor network. It collects the information captured by all the sensors and sent by the sensor stations. The central server includes the Esper CEP engine that applies the patterns designed to control the malfunctions of the clocks in the sensor stations. Its main functionalities are to send resets in case of errors detected in the sensor stations and report the results of the patterns. These results are sent to the stations every 60 min and they represent an average of the differences that they have suffered during that time lapse.

In Figures 4–6, we show the automata that represent the behaviour of a sensor, a sensor station and the central server, respectively.

In addition to specify the functional behaviour of the components of the network, we need to define certain (time) constraints associated with both action transitions and states of the automata. On the contrary to conventional constraints, fuzzy constraints give us a satisfaction grade in the interval $[0, 1]$ about the confidence that we have in an assessment. Due to the fact that we deal with time, it is not feasible to work with strict categorizations of the truth or falsehood of an evaluation. For this reason, we consider a range of possible values: the larger the satisfaction grade, the more confidence we have in the assessment. If any of these conditions are represented by the value *True*, it is not necessary to consider any constraint in that transition. These transitions are always available to be performed. For example, the constraint $\overline{x - y \leq 10}^{1}$ establishes a restriction over the difference between the values of the variables x and y. If the difference is lower than or equal to 10, with a fuzzy limit of 1, the constraint fulfills with a satisfaction grade of 1. However, if the difference is, for example, 10.6, then the grade of confidence is lower. If this constraint is associated with a transition, then the transition can be triggered if the constraint is satisfied by the values of x and y. Similarly, if the constraint is associated with a state of the automaton, the automaton can remain at this state while the constraint fulfills.

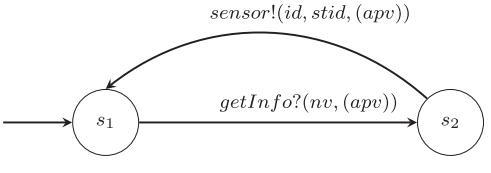

Figure 4. Sensor automaton.

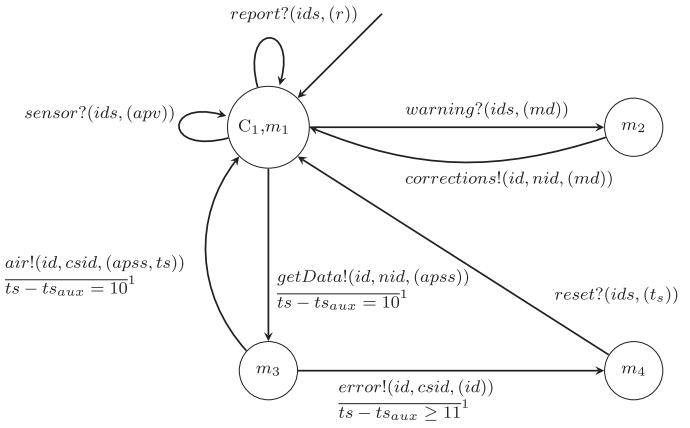

Figure 5. Sensor station automaton.

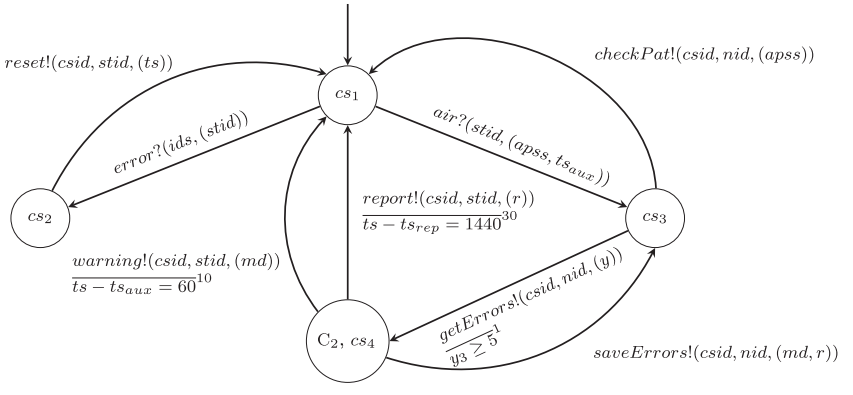

Figure 6. Central server automaton.

4.2. A fuzzy logic approach for processing sensor time malfunctions and failures

In this section we describe the data processing approach that we apply to analyse the information observed in the sensor network. This process follows the next steps:

- *Step 1*: time information. The information related to each sensor station during 60 min is stored in a 24-tuple. Specifically, we store the average of the time differences, in absolute value, between the local time when information is sent to the central server and the server time when the information is received.
- *Step 2*: correctness functions. We check the mismatches that might have been produced in each time slot. These functions are designed to detect both *malfunctions* and *failures* in the behaviour of the stations. The former arise when the differences, in average, are between 5 and 10 min, in a fuzzy sense. The latter correspond to an average fuzzily greater than 10 min. Specifically, these fuzzy order relations are the following:
- *Step 3*: *t*-norm estimation. We apply a triangular-norm in order to evaluate the *conjunction* of the grades of satisfaction previously obtained. Specifically, we apply the

Hamacher *t*-norm to the hours of a day of each station. Since *t*-norms are associative, it is enough to define its binary version and apply it to all the data (in our case, 24 values).
- *Step 4*: checking correctness. We impose a minimal grade of satisfaction for considering that the behaviour of a station during a day is correct. The established threshold is 0.75, so that we can discard some errors due, for example, to the overload of the network where the data are transferred.

4.3. Event patterns for malfunction and failure station detection

In order to process the real-time sensing data, we have implemented in Esper EPL (Esper, 2017) a set of event patterns to be deployed in the Esper CEP engine. This is the software responsible for detecting situations of interest (event patterns) from input data (simple events). These simple events share a unique format in which the values for pollutants $PM_{2.5}$, PM_{10}, CO, O_3, NO_2 and SO_2 are registered for every sensor station at a particular location. In addition, the timestamp in which these pollutant values were sensed (*stationTs*) and the timestamp in which these data have reached the server (*currentTs*) are also registered into simple events. We have implemented the following event patterns:

- *TimeDiffAvg*. This pattern computes the 1-hour average of the difference, in absolute value, between *currentTs* and *stationTs* by sensor station. As a result, a complex event *TimeDiffAvg* is generated with the following information: the current timestamp in seconds, *timestamp*, the *stationId* and the observed time difference, *timeDiff*. This pattern is used to carry out the Step 1.
- *Malfunction*. This pattern detects when a station clock is slightly advanced/delayed. This happens when the 1-hour average of the difference in absolute value between *currentTs* and *stationTs* is fuzzily greater/smaller than or equal to 5/10 min, respectively. The user-defined function *malfunctionFuzzy(timeDiff)* has been implemented to compute the pseudo fuzzy limits for malfunction detection. This pattern is used to perform the Step 2.
- *Failure*. This pattern detects when a station is not working. This happens when the 1-hour average of the difference in absolute value between *currentTs* and *stationTs* is greater than 10 min. The user-defined function *failureFuzzy(timeDiff)* has been implemented, as explained before, to compute the pseudo fuzzy limits for failure detection. This pattern is used to reach the goal of the Step 2.
- *MalfunctionTNorm*. This pattern calculates the *t*-norm of Hamacher with the hours of a day for each station in which a *Malfunction* complex event has been detected. This pattern is part of the implementation of Step 3.
- *FailureTNorm*. This pattern computes the *t*-norm of Hamacher with the hours of a day for each station in which a *Failure* complex event has been detected. It is also associated with the previously mentioned Step 3.
- *MalfunctionReportPattern*. This pattern checks a minimal satisfaction grade for considering as correct the behaviour of a station during a day in which a *MalfunctionTNorm* complex event has been detected. This pattern is associated with Step 4.
- *FailureReportPattern*. This pattern checks a minimal satisfaction grade for considering as correct the behaviour of a station during a day in which a *FailureTNorm* complex event has been detected. This pattern is also related to Step 4.

```
@Name('MalfunctionTNorm')
insert into MalfunctionTNorm
select current_timestamp() as timestamp, stationId as stationId,
  window(*).aggregate(1, (result, malfunction) =>
        (case when result = 0.0 and malfunction.delta = 0.0 then 0 \newline
         else (result * malfunction.delta) / ((result + malfunction.delta) -
              result * malfunction.delta) end)) as t,
  window(*).aggregate(0, (result, malfunction) =>
        (case when malfunction.delta = 0.0 then result + 1\newline
         else result end)) as zeroDeltaNum
from Malfunction.win:time_batch(86400)
group by stationId
```

Figure 7. Esper EPL implementation of the *MalfunctionTNorm* event pattern.

As an example, Figure 7 shows the Esper EPL implementation of the *MalfunctionTNorm* event pattern.

5. Analysis of the results

This study aims to test some hypotheses:

(1) CEP technology is useful to automatically detect situations of interest in real-time, while the data transferred across a sensor network is processed, in contrast to other traditional approaches that, in order to analyse the data, require to store the information previously.
(2) Fuzzy reasoning allows us to deal with imprecision in the collected information
(3) A set of event patterns based on fuzzy logic can be defined to automatically detect potential (temporary) malfunctions and failures in the sensor clocks.
(4) Our approach can take appropriate decisions derived from the collected results as well as generating alarms (complex events) at the right time.

In order to do it, we analyse and report the obtained results from our experiments in which we have applied our approach to the air quality sensor network introduced in Section 4. We focus on the information corresponding to sensors stations 44, 49 and 55, and the central server during December 2016. We have chosen these three stations because they show some of the problems that we were looking for. Specifically, the sensor station 44 presents both malfunctions and failures, while 49 and 55 present many malfunctions but few failures. The event pattern implementation and the results of the whole network are available at http://dx.doi.org/10.17632/ccsn9xj4y8.2.

Figure 8 shows *Malfunction* complex events detected in these stations during December 2016. The events located between 300 and 600 s represent confirmed malfunctioning stations, since this is the valid limit considered for detecting this type of situations. However, the data above 600 s or below 300 s not always correspond to malfunction detections. It depends on the minimal satisfaction grade that we establish to consider as correct the behaviour of a station during a day. Table 1 shows the precise number of confirmed malfunctioning stations detected per day. For example, although there are

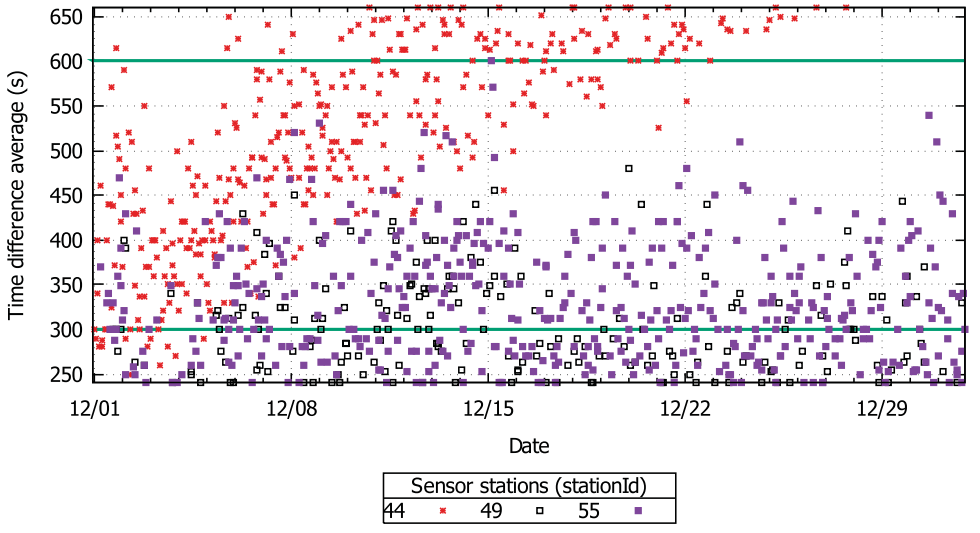

Figure 8. Malfunction sensor station detection during December 2016.

Table 1. Malfunction and failure detections for stations 44, 49 and 55 during December 2016.

Day	Malfunction			Failure		
	44	49	55	44	49	55
1	18	3	10	1	0	0
2	20	2	5	1	0	0
3	20	1	2	0	0	0
4	23	1	3	0	0	0
5	21	4	12	3	0	0
6	21	6	13	3	0	0
7	22	6	11	2	0	0
8	20	3	6	4	0	0
9	20	5	9	4	0	0
10	17	1	8	8	0	0
11	12	9	13	13	0	0
12	10	12	13	14	0	0
13	9	6	18	15	0	0
14	8	6	14	17	0	0
15	7	6	12	18	1	1
16	7	2	5	19	0	0
17	4	1	11	21	0	0
18	4	0	8	20	0	0
19	2	2	11	22	0	0
20	2	4	9	24	0	0
21	3	1	13	23	0	0
22	2	2	12	23	0	0
23	0	3	8	24	0	0
24	0	0	8	24	0	0
25	0	3	13	24	0	0
26	0	2	9	24	0	0
27	0	6	10	24	0	0
28	0	2	11	24	0	0
29	0	5	6	23	0	0
30	0	3	11	24	0	0
31	0	1	10	24	0	0

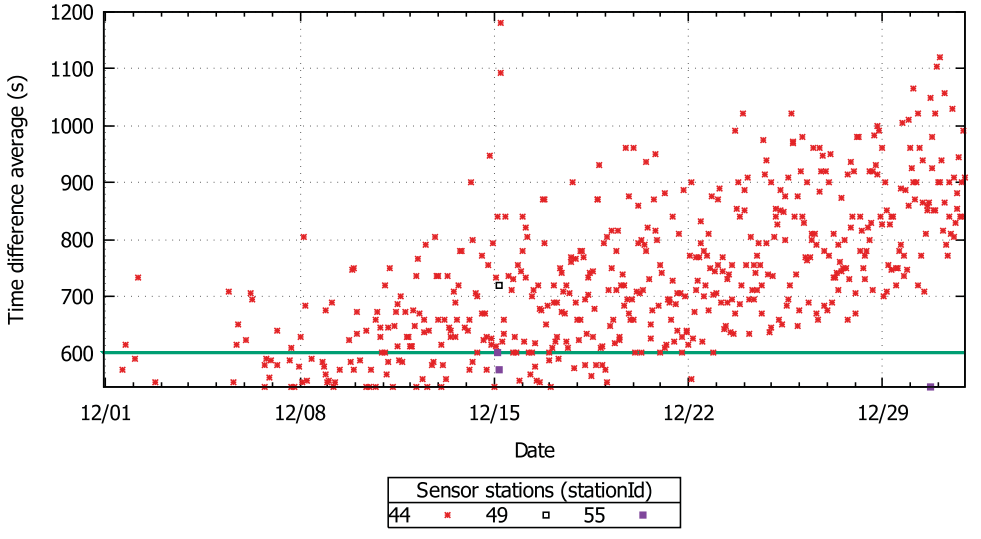

Figure 9. Failure sensor station detection during December 2016.

some events associated with the 44 station above 600 s registered from December 23, none of them has been classified as a possible error.

Figure 9 depicts every *Failure* complex event detected for these stations. In this case, the events located above 600 s represent confirmed failure stations, being those below 600 s non-confirmed failure detections. While station 44 has worked improperly during the whole month, a unique confirmed failure has been registered for stations 49 and 55. It is worth noting that although there are two additional detection cases for station 55, these are not classified as errors by the *FailureReport* pattern.

As a conclusion, we can claim that our approach, based on CEP and fuzzy logic, is useful to automatically detect real-time clock malfunctions and failures in a sensor network. This detection can be followed by a notification concerning the affected stations so that the observed errors can be fixed as soon as possible. For instance, station 44 should have been fixed before December 15, given the high number of failures detected. However, this is something out of our control: the regional government gave us access to the data but we have no power to fix the detected problems.

6. Conclusions

We have analysed the results of observing a sensor network located in Andalusia (South of Spain) during one whole month. This network is conformed by 61 sensor stations, each of them receiving data from six sensors, and a central server that are devoted to the study of air quality. The first complication that we encountered was to process a huge amount of data, in real time, to find errors. The second problem was to appropriately define when a certain error had been found. In this paper we have concentrated on the results concerning the accuracy of the clocks associated with all the components of the network. The clocks used in the stations presented some important lacks of synchronization with respect to the central server clock: this fact provokes the inaccurate evaluation of

the events. In order to avoid this problem, we looked for formalisms to represent both the expected behaviour of a system with uncertain information (as the measurement of time) and the patterns that would catch the occurrence of unexpected data. For the first task, we designed a variant of timed automata where fuzzy constraints control time values. Second, we considered the CEP technology to define patterns to match the observed values. The analysis of data from this scenario has shown the benefits of using formal representations of the system and properties.

We have several lines for future work. First, we would like to improve the design of the data processing by including more precise patterns. These additional patterns should provide information about the specific instant when the station has suffered a lack of synchronization and analyse the possible environmental and atmosphere reasons that provoked them. Second, we would like to apply our combined fuzzy automata and CEP approach to other application domains where CEP has already been successfully used. For example, we are considering healthcare and network security and we will take as initial step our previous work (Boubeta-Puig, Ortiz, & Medina-Bulo, 2014, 2015; Macià, Valero, Díaz, Boubeta-Puig, & Ortiz, 2016).

Acknowledgements

The first author would like to thank the hospitality of the *Design and testing of reliable systems* research group, at Universidad Complutense de Madrid, during his stay when this research was carried out.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Spanish MINECO/FEDER project DArDOS under grant nos. TIN2015-65845-C3-1-R and TIN2015-65845-C3-3-R; the Comunidad de Madrid project SICOMORo-CM under grant no. S2013/ICE-3006; the University of Cádiz under grant no. PR2016-032.

Notes on Contributors

Juan Boubeta-Puig received the Ph.D. degree in Computer Science from the University of Cádiz (UCA), Spain, in 2014. Since 2009, he has been an Assistant Professor with the Department of Computer Science and Engineering at UCA. His research focuses on the integration of complex event processing in event-driven service oriented architectures, the Internet of Things, and model-driven development of advanced user interfaces. He was honoured with the Extraordinary Ph.D. Award from UCA and the Best Ph.D. Thesis Award from the Spanish Society of Software Engineering and Software Development Technologies.

Mario Bravetti is an Associate Professor at the Computer Science and Engineering Department of University of Bologna. He is also member of the FOCUS (FOundations of Component-based Ubiquitous Systems) team which is part of the INRIA Sophia Antipolis - Méditerranée research center. His research activity spans from formal description and analysis of concurrent/distributed systems based on mathematical and probabilistic methodos to more applicative topics such as service oriented and cloud computing. He was winner of the award for best italian PhD thesis in theoretical

computer science, assigned by the Italian Chapter of the European Association for Theoretical Computer Science.

Luis Llana is an Associate Professor in the Computer Systems and Computation Department, Complutense University of Madrid (Spain). He obtained his MS degree in Mathematics 1991 and his PhD in the same subject in 1996. His main research interest fields are formal methods and testing techniques. Currently he is opening new research fields such as artificial vision and e-learning.

Mercedes G. Merayo received her Ph.D. in Computer Science from Universidad Complutense de Madrid, Spain, in 2009. She holds an Associate Professor position in the Computer Systems and Computation Department at the same University. She has published more than 60 papers in refereed journals and international venues. She regularly serves in the Program Committee of conferences such as SEFM, ICTSS or QRS. Dr. Merayo has co-chaired QSIC 2011, SEFM 2013, ICTSS 2014 and SAC-SVT 2017 among others. Her current research interests include model based testing, distributed testing, asynchronous testing, mutation testing and timed extensions in formal testing.

ORCID

Juan Boubeta-Puig http://orcid.org/0000-0002-8989-7509
Mercedes G. Merayo http://orcid.org/0000-0002-4634-4082

References

Alur, R., & Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, *126*, 183–235.

Andrés, C., Llana, L., & Núñez, M. (2011). *Self-adaptive fuzzy-timed systems*. Paper presented at the 13th IEEE congress on evolutionary computation (CEC' 11) (pp. 115–122). Washington, DC: IEEE Computer Society.

Bouteta-Puig, J., Camacho, A., Llana, L., & Núñez, M. (2017). *A formal framework to specify and test systems with fuzzy-time information*. Paper presented at the 14th int. conf. on artificial neural networks, iwann'17, lnai. Berlin: Springer.

Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2014). A model-driven approach for facilitating user-friendly design of complex event patterns. *Expert Systems with Applications*, *41*(2), 445–456.

Boubeta-Puig, J., Ortiz, G., & Medina-Bulo, I. (2015). MEdit4CEP: A model-driven solution for real-time decision making in SOA 2.0. *Knowledge-Based Systems*, *89*, 97–112.

Chandy, K., & Schulte, W. R. (2010). *Event processing: Designing IT systems for agile companies*. New York, NY: McGraw-Hill.

Doostfatemeh, M., & Kremer, S. C. (2005). New directions in fuzzy automata. *International Journal of Approximate Reasoning*, *38*(2), 175–214.

Doostfatemeh, M., & Kremer, S. C. (2006). *General fuzzy automata, new efficient acceptors for fuzzy languages*. 2006 ieee int. conf. on fuzzy systems (fuzz-ieee'06) (pp. 2097–2103). Washington, DC: IEEE Computer Society.

EsperTech. (2017). Esper. Retrieved from http://www.espertech.com/esper/documentation.php.

Etzion, O., & Niblett, P. (2011). *Event processing in action*. Greenwich, CT: Manning.

Event Processing Glossary – Version 2.0. (2011). Retrieved from http://www.complexevents.com/wp-content/uploads/2011/08/EPTS_Event_Processing_Glossary_v2.pdf.

Fernández-Vilas, A., Pazos-Arias, J. J., & Díaz Redondo, R. P. (2002). *Extending timed automaton and real-time logic to many-valued reasoning*. Paper presented at the 7th int. symposium on formal techniques in real-time and fault-tolerant systems (ftrtft'02), lncs 2469 (pp. 185–204). Berlin: Springer.

Lamport, L. (2015). Who builds a house without drawing blueprints?. *Communications of the ACM*, *58* (4), 38–41.

Luckham, D. (2002). *The power of events: An introduction to complex event processing in distributed enterprise systems*. Boston, MA: Addison-Wesley.

Luckham, D. (2012). *Event processing for business: Organizing the real-time enterprise*. Hoboken, NJ: John Wiley & Sons.

Macià, H., Valero, V., Diaz, G., Boubeta-Puig, J., & Ortiz, G. (2016). Complex event processing modeling by prioritized colored petri nets. *IEEE Access*, *4*, 7425–7439.

Mensch, S. I., & Lipp, H. M. (1990). *Fuzzy specification of finite state machines*. Paper presented at the 1st European design automation conference (euro-dac'90) (pp. 622–626).

Mordeson, J. N., & Malik, D. S. (2002). *Fuzzy automata and languages: Theory and applications*. London/Boca Raton, FL: Chapman & Hall/CRC.

Mraz, M., Lapanja, I., Zimic, N., & Virant, J. (1999). *Fuzzy numbers as inputs to fuzzy automata*. Paper presented at the 18th international conference of the North American Fuzzy Information Processing Society (nafips'99) (pp. 453–456). Washington, DC: IEEE Computer Society.

Wee, W. G., & Fu, K. S. (1969). A formulation of fuzzy automata and its application as a model of learning systems. *IEEE Transactions on Systems Science and Cybernetics*, *5*(3), 215–223.

WHO Media Centre (2016). *Ambient (outdoor) air quality and health*. Retrieved from http://www.who.int/mediacentre/factsheets/fs313/en/.